A Novel Task Partitioning Method for Multi-core Processor Based-on Cohesion and Coupling

Jianchun Jiang
College of Automation, Chongqing University of Posts and Telecommunications Chongqing 400065, China
Email: dennis_jjc@yahoo.com.cn

Suhua Zeng
School of Computer Science and Technology, Chongqing University of Posts and Telecommunications Chongqing 400065, China
Email: zengsh@cqupt.edu.cn

***Abstract*—Task partition is a NP-Hard problem on multi-core processor. To achieve a good MCT (maximum completion time) goal, this paper presents a novel task partitioning method for multi-core processor based on the cohesion and coupling properties of tasks. First, the relations between the cohesion and coupling are analyzed and the computation method of coupling cost based on cohesion is designed with the task scheduling requirements considered. Then, aiming to the problem that the MCT (maximum completion time) is different when the tasks' coupling is changed and the tasks are rescheduled during the task partitioning process, an evaluating method of MCT with the considering of task scheduling requirements is designed. Last, this paper compartmentalizes tasks based on the MCT to make it minimum. This method reduces the processing of the combinatorial optimization problem in task partitioning process, and builds the relation between task partition and task scheduling. The experimental results testify this method available.**

***Index Terms*—multi-core processor, Task Partition, Cohesion, Coupling**

I. INTRODUCTION

The aim of task partitioning is to obtain a task set, which can be scheduled on the most suitable core, to make system achieve the lowest power consumption, balancing load, the minimum MCT, the highest executing efficiency and the efficient usage of resources. Task partitioning is the precondition of task scheduling. But in the most documents about task scheduling researches of multi-core system or distributed system, such as Ref. [1] and Ref. [2], which mainly stressed how to schedule task and neglected the effect of task partitioning, and the scheduling results are not optimal. The research about task partitioning is necessary, especially in multi-core processor system.

The task partitioning on multi-core processor is either related to task function, the feature of processor and resources, or related to the task scheduling. So how to partition tasks effectively is a combinatorial optimization problem, which is also proved a NP-Hard [3] problem. After analyzing the feature of heterogeneous multi-core processor, the main problems of task scheduling are studied firstly, and then the current existent solving methods, afterward a cohesion and coupling based task scheduling way to meet the minimum MCT requirement of system is presented.

The remainder of this paper is organized as follows: section 2 states the current main research work. In the section 3, the relations of cohesion and coupling are analyzed, and the computing way of the coupling based on cohesion is designed. Authors discuss and design the task partitioning method in section 4. The method feasibility and validity are tested in section 5. Last, authors conclude the task partitioning method and propose the next work.

II. RELATED WORK

Sathish Gopalakrishnan and Marco Caccamo[3] proposed a Fully Polynomial-Time Approximation Scheme (FPTAS) to solve replication problem for the heterogeneous multiprocessor task partitioning, namely how to determine a mapping of recurring tasks upon a set consisting of different processing units, in which all tasks meet their timing constraints and no two replicas of the same task are assigned to the same processing unit. Nathan Fisher，James H. Anderson and Sanjoy Baruah[2] presented an efficient algorithm for solving the memory constrained of the multiprocessor partitioning, in which the problem of task partitioning was formalized in a manner that was cognizant of both memory and processing capacity constraints. Aiming to solve task partitioning problem upon preemptive heterogeneous multiprocessor platforms, Sanjoy K. Baruah[5] designed a polynomial-time approximation algorithm, which is a near-optimal solution for the integer linear programming problem to minimize the schedule makespan and meet the deadline goal of the schedule. Reference [6] proposed a clustering framework, along with a flexible multi-purpose clustering algorithm that initiated task clustering at the functional level based on dynamic profiling information, to take full advantage of the multi-core processor efficiency. Zheng Kai[7] compared the performance between the functional partitioning and

doi:10.4304/jcp.8.5.1247-1254

the partitioning by data for multimedia streaming application, and presents the solution how to achieve the lower power consumption and the better performance.

Ming Wu and Xian-He Sun[8] investigated three partition policies to obtain the better performance in non-dedicated grid computing environment with resource sharing, and analyzed the partitioning methods based on memory-only, CPU-memory and Shared resources. Reference [9] presented an optimization performance model for the distribution of computations over heterogeneous computers, in which the processor heterogeneity, the heterogeneity of memory structure, and the memory limitations were all considered, and designed the efficient formulated method for the model. Reference [10] proposed a novel task partitioning algorithm for intra-task DVS which partitions a given task so that the DVS can be applied more effectively with the minimum number of voltage switching. Fang-Wu Yao and Zhao-Cai Lu[11] proposed an immune task partitioning algorithm of alterable coefficient of parameters driven hardware-software partitioning methods based on the consideration of the characteristics of dynamically reconfiguration. Ants algorithm, Heuristic Search Methods, Tabu Search algorithm were used to solve the task partitioning problem in software and hardware co-design system in articles [12-14].

All these research mainly analyzed the relation between partition and system constraint, such as power consumption, efficiency, the minimum makespan, memory space, processor feature, and so on, and designed the partitioning approaches without adequately considering the self-task characters and the scheduling requirements. In the multi-core processor, task complete time is one of the most important targets, and it relates with the task scheduling requirement besides the inter-task coupling attribute. So studying the partitioning method based on task attribute is very significant.

III. The Cohesion And Coupling Of Tasks

A. The Relation between Cohesion and Coupling

There are a lot of complex interdependences among different modules, such as sharing devices and program modules, the different sequence requirements, data communication. In general, it is expressed in the forms of communication, mutex, synchronization, and used to be measured by the degree of coupling, which is used to describe the strength of relationship between two modules. The coupling can be categorized into Data Coupling, Stamp Coupling, Control Coupling, External Coupling, Common Coupling and Content Coupling according to the coupling degree [15]. Meanwhile, the different modules possess different inherent feature, which can be expressed by the cohesion. The cohesion can also be measured off using Functional Cohesion, Sequential Cohesion, Communicational Cohesion, Temporal Cohesion, Logical Cohesion and Coincidental Cohesion. The software modules must be designed with high cohesion and low coupling.

In the task partitioning process, the cohesion and coupling are the two major measurable indices. The cohesion is used as the clustering condition, and the coupling is used to measure performance of the partitioning results in every stage. As results from analyzing, the different module with obvious cohesion is corresponded with the one main different coupling. For example, the coupling between Functional Cohesion modules mainly behaves as Data Coupling, and between the different Sequential Cohesion modules with the same functional cohesion, the coupling is mainly Stamp Coupling. On the other hand, the same preceding cohesion modules with different succeeding cohesion should be measured with the nearest same preceding coupling. For instance, the same Sequential Cohesion modules with different Communicational Cohesion should be evaluated with the Control Coupling corresponding to Sequential Cohesion. The others like the same way. The Fig.1 shows the coupling between different cohesion modules.

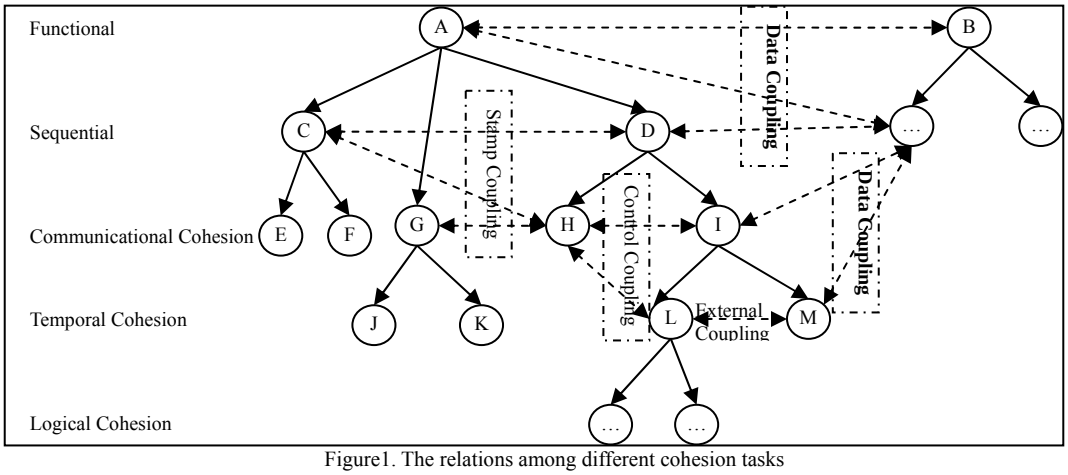

Figure1. The relations among different cohesion tasks

During partitioning, modules should satisfy the low coupling and high cohesion, and be partitioned according to Functional Cohesion, Sequential Cohesion, Communicational Cohesion, Temporal Cohesion, and should avoid being clustered with Logical Cohesion and Coincidental Cohesion.

B. Cohesion-based Coupling Computing Method

The previous analyzing results show there are different couplings between the different cohesion modules. Therefore, while the modules are running, the system will spend more time, named coupling cost, to deal with such as communication, synchronization, mutex, which are brought owing to the coupling between them. The coupling cost not only is related to partitioning, but also is one of the references of task scheduling and dispatch. Therefore, how to compute coupling is very significant. This section proposes a coupling computing method based on cohesion weight.

1) Task cohesion coding

According to above results, there are different coupling between different cohesion modules. In order to partition tasks, the task module can be described with different cohesion code. The formula (1) shows the basic format of the code.

$$Cohesion_i(fun_i, seq_i, com_i, tem_i, log_i, coi_i) \quad (1)$$

fun_i, seq_i, com_i, tem_i, log_i and coi_i are in turn the functional, sequential, communicational, temporal, logical and coincidental cohesion. $fun_i \in [1, k_1]$, $seq_i \in [0, k_2]$, $com_i \in [0, k_3]$, $tem_i \in [0, k_4]$, $log_i \in [0, k_5]$, $coi_i \in [0, k_6]$, and k_1 - k_6 express the sorts of each cohesion.

2) The coupling computing method based on cohesion

With the difference of the coupling nature, the coupling cost generated during scheduling is different, and the total complete time is also different. So it is indispensable to design a method to calculate the coupling cost. From small to large, the weight parameters are $w_{content}$, $w_{common}, w_{external}, w_{control}, w_{stamp}, w_{data}$. The weight can be set in proportion with the analyzed results from Ref. [15], and the base $w_{content}$ is equal 1. Because the coupling is related closely to the cohesion, supposed that the numbers of the coupling times for different coupling nature inter-task are [n_1, n_2, n_3, n_4, n_5, n_6], the coupling value can be calculated as formula (2) according to cohesion code of task.

$$\begin{aligned} coup_{i,j} = {} & \overline{(fun_i \oplus fun_j)} \cdot n_1 \cdot w_{content} + \overline{(fun_i \oplus fun_j)} \cdot ((\underline{seq_i \oplus seq_j}) \cdot n_2 \cdot w_{i,j,common} \\ & + \underline{(seq_i \oplus seq_j)} \cdot ((com_i \oplus com_j) \cdot n_3 \cdot w_{i,j,external} + \underline{(com_i \oplus com_j)} \cdot ((tem_i \oplus tem_j) \cdot n_4 \cdot w_{i,j,control} \\ & + (tem_i \oplus tem_j) \cdot ((log_i \oplus log_j) \cdot n_5 \cdot w_{i,j,stamp} + \overline{(log_i \oplus log_j)} \cdot (coi_i \oplus coi_j) \cdot n_6 \cdot w_{i,j,data}))) \end{aligned} \quad (2)$$

In formula(2), the cohesion codes in the corresponding position XOR each other. Firstly, the functional cohesion code XOR, if the codes are same, the coupling is decided by the result of the next position cohesion code XOR. Otherwise, the coupling value is decided by the coupling corresponding to the cohesion and the coupling times regardless of the next XOR results. The others are similar. For example, there are two tasks, Task1[1,0, 0,1,2] and Task2[1,2,5,4,6], whose coupling is decided by sequential cohesion; but for Task3[2,3,2,4,6] and Task4[2,3,2,5,3], whose coupling is decided by Temporal Cohesion.

One task may couple with multiple tasks, and there may be multiple couple between two tasks. The coupling strength is decided by the coupling nature and the times of coupling. The vector $coup_i = (coup_{i,1}, coup_{i,2}, \cdots, coup_{i,j}, \cdots, coup_{i,n})$ is used to express the coupling of one task with others, and the self-coupling $coup_{i,i} = 1$. In partitioning process, the threshold value $\mu \le 1$ is used as a coupling boundary value whether task is divided. If coupling is larger than μ, the coupling of the task with other task is too strong, the task need to combine with the other task.

IV. The Partitioning Method Based On Cohesion And Coupling

A. The Target Function of Task Partitioning

The main partitioning targets are the highest efficiency, the minimum complete time, the lowest power consumption, and so on. This article main solve the minimum MCT problem in multi-core processor system. A set of modules (namely a large task) can be divided into k_i subtask, and then the total complete time of the tasks is depended the executing time (ET), scheduling cost and task coupling cost while cores are certain. The complete time is showed in formula (3).

$$T_{i,max} = \max_{1 \le u \le h_i}(T_u + T_{u,coup} + T_{u,sch}) \quad (3)$$

T_u is the ET of all task dispatched on core u, $T_{u,coup}$ is the coupling cost on it, and $T_{u,sch}$ is the scheduling cost, h_i is the number of cores that can run task i.

B. The Partitioning Method Design based on Cohesion and Coupling

In this paper, the partitioning method is designed to reduce the complete time of system which is related directly to the ET, scheduling cost and the coupling cost. So the computing approach of complete time will be analyzed and designed in this section.

1) Coupling cost

Based on the formula (2), t_{coup} is used to express the basic coupling cost, and the inter-task coupling cost can use formula (4) to calculate.

$$t_{i,j,coup} = c_{i,j,coup} \times coup_{i,j} \times t_{coup} \quad (4)$$

In formula (4), $c_{i,j,coup}$ is core's effect parameter. If the two coupling tasks are all dispatched on the same core, $c_{i,j,coup} = 1$. Otherwise, $c_{i,j,coup} > 1$. No considering the difference of cores which coupling task running on, $t_{i,j,coup} = t_{j,i,coup}$.

2) Scheduling cost

Besides related to the number of tasks, scheduling cost is also connected with the position of scheduling. $c_{j,sch}$ is the difference parameter between inter-cores and intra-cores. When a task is scheduled in the same core, $c_{j,sch}=1$. Otherwise, $c_{j,sch}>1$. Considering the basic scheduling cost is t_{sch}, the task scheduling cost is showed as formula (5).

$$t_{j,sch}=c_{j,sch}t_{sch} \tag{5}$$

3) Task parallelizing parameter

In multi-core processor, the complete time of tasks is also related to the degree of tasks parallelization. The parallelization means that multiple tasks can be executed parallel after partitioned. If tasks are coarse granularity, the higher parallel degree can not be obtained. some tasks can be only executed orderly, and the efficiency reduces. So the parallelization should be considered in partitioning process.

The whole parallelization is related with the cores running them. Generally, the degree of parallelization increases with the number of cores. Supposed task i could be divided into k_i subtasks, and there are h_i cores can run them. The parallel parameter can be calculated as formula (6).

$$p_i=\begin{cases}1 & h_i\geq k_i\\ \dfrac{h_i}{k_i} & h_i<k_i\end{cases} \tag{6}$$

When $h_i\geq k_i$, all subtasks can be executed simultaneously in theory. So $p_i=1$. But when $h_i<k_i$, the k_i subtasks are dispatched on the h_i cores. Some tasks are leaved for no sufficient cores. The average value of parallelization is proposed.

4) The estimate of task complete time

Despite the effecting parameters of the task complete time have been analyzed previously in the partitioning stage, it is unknown which cores the task will be dispatched onto in scheduling stage. So the executing time, the scheduling cost and the coupling cost that happened in scheduling stage must be evaluated.

Suppose t_i is the ET of task i, if task i is partitioned into k_i subtasks whose ET are $t_{i,1}$、$t_{i,2}$、……、t_{i,k_i}. In the light of the LETFS (Longest Executing Task First Scheduling) rule, the max ET of all subtasks can be computed as formula (7).

$$T'_u=\frac{\bar{t}_{i,j}}{p_i}+\frac{k_i p_i-1}{k_i-1}\cdot(\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j}) \tag{7}$$

In formula (7), $\bar{t}_{i,j}=\frac{1}{k_i}\sum_{j=1}^{k_i}t_{i,j}$ is the average ET of all subtasks. In term of the requirement of load balance, the max ET is limited not over 2 times of average ET, namely $\max(t_{i,j})\leq 2\bar{t}_{i,j}$.

The feasibility of formula (7) is proofed as follows.

Proof:

a) if $h_i\geq k_i$, $p_i=1$, which means tasks can be executed simultaneously, so formula is available.

b) when $h_i<k_i$, all the subtasks of task i are impossible to be executed simultaneously. But the formula should accurately calculate the max complete time, and make the max complete time accord with the limitation of executing time. The evaluated complete time should not exceed the max consecutive processing time of the task i alone.

Learning from the formula (6), $p_i<1$. So with h_i becoming smaller, p_i becomes smaller and the degree of parallelization will be also declined. When $\bar{t}_{i,j}/p_i$ become bigger, the T'_u will become bigger if $\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j}$ no change. In other words, the max executing time is increasing, and it matches with the change trend of parallelization. When task i is divided into k_i subtasks which are allocated on h_i cores to run, T'_u should satisfies the following condition.

$$\begin{aligned}&\frac{\bar{t}_{i,j}}{p_i}+\frac{k_i p_i-1}{k_i-1}\cdot(\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j})\\ &=\frac{k_i\bar{t}_{i,j}}{h_i}+\frac{k_i p_i-1}{k_i-1}\cdot(\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j})\\ &=\frac{k_i\bar{t}_{i,j}}{h_i}+\frac{h_i-1}{k_i-1}\cdot(\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j})\\ &\leq k_i\bar{t}_{i,j}\\ &=\sum_{j=1}^{k_i}t_{i,j}\end{aligned}$$

Because $(\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j})\leq\bar{t}_{i,j}$, So,

$$\begin{aligned}\frac{k_i\bar{t}_{i,j}}{h_i}+\frac{h_i-1}{k_i-1}\cdot(\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j})&\leq\frac{k_i}{h_i}\bar{t}_{i,j}+\frac{h_i-1}{k_i-1}\bar{t}_{i,j}\\ &=\bar{t}_{i,j}(\frac{k_i}{h_i}+\frac{h_i-1}{k_i-1})\end{aligned}$$

Because $1\leq h_i\leq k_i$, setting $h_i=k_i-n$, $n=1,2,\cdots,k-1$,

$$\begin{aligned}&\frac{k_i}{h_i}+\frac{h_i-1}{k_i-1}\\ &=\frac{k_i}{k_i-n}+\frac{k_i-n-1}{k_i-1}\\ &=1+\frac{n}{k_i-n}+1-\frac{n}{k_i-1}\\ &=2+\frac{n}{k_i-n}-\frac{n}{k_i-1}\end{aligned}$$

If $p_i\downarrow$, $h_i\downarrow$, $T'_u\uparrow$, and $\frac{k_i}{h_i}\geq\frac{h_i-1}{k_i-1}$, So

$$\begin{aligned}&\frac{k_i\bar{t}_{i,j}}{h_i}+\frac{h_i-1}{k_i-1}\cdot(\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j})\\&\le k_i\bar{t}_{i,j}\\&=\sum_{j=1}^{k_i}t_{i,j}\end{aligned}$$

The previous proofs have proved the ET of subtasks after parallelized would exceed the consecutive processing time.

On the other hand, the complete time after parallelized should be larger than the average time of all subtasks about the executing cores. Proofs are follows.

$$\begin{aligned}&\frac{\bar{t}_{i,j}}{p_i}+\frac{k_i p_i-1}{k_i-1}\cdot(\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j})\\&=\bar{t}_{i,j}\cdot k_i/h_i+\frac{k_i p_i-1}{k_i-1}\cdot(\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j})\\&\ge\bar{t}_{i,j}\cdot k_i/h_i\\&\ge\bar{t}_{i,j}\end{aligned}$$

So the formula (7) is valid.

At the same time, the scheduling cost is related to the number of tasks, so the total scheduling cost of all subtasks is $k_i t_{sch}$, and the average cost on each core is $T'_{u,sch}=k_i t_{sch}/h_i=t_{sch}/p_i$. And the coupling cost of all tasks in one core can be expressed by $\sum_{j,k=1}^{k_i}t_{j,k,coup}$, and $j\neq k$. If the coupling differences between inter-core and intra-core are ignored, the average coupling cost on every core is $T'_{u,coup}=\frac{1}{h_i}\sum_{j,k=1}^{k_i}t_{j,k,coup}$.

When k_i subtasks of the task i are dispatched on h_i cores, considering load balance, the differences of the inter-core scheduling and coupling cost are not very obviously. The scheduling cost and the coupling cost can be substituted by average values above proposed. So the max complete time after each partitioning stage can be computed as formula (8).

$$\begin{aligned}T'_{i,max}&=T'_u+T'_{u,sch}+T'_{u,coup}\\&=\frac{\bar{t}_{i,j}}{p_i}+\frac{k_i p_i-1}{k_i-1}\cdot(\max_{j\in[1,k_i]}(t_{i,j})-\bar{t}_{i,j})+\frac{t_{sch}}{p_i}+\frac{1}{h_i}\sum_{j,k=1}^{k_i}t_{j,k,coup}\\&=\frac{\bar{t}_{i,j}+t_{sch}}{p_i}+\frac{k_i p_i-1}{k_i-1}\cdot(\max(t_{i,j})-\bar{t}_{i,j})+\frac{1}{h_i}\sum_{j,k=1}^{k_i}t_{j,k,coup}\end{aligned}\quad(8)$$

From the formula we can conclude that the complete time is related nearly to the degree of parallelization. When the number of subtasks of the task i, $T'_u\downarrow$, $T'_{u,sch}\uparrow, T'_{u,coup}\uparrow$; and when $p_i\uparrow$, $\max(t_{i,j})\downarrow$, but the scheduling cost and the coupling cost will be increased. So it is necessary to weigh the degree of parallelization and make $T_{i,max}$ minimum in partitioning process.

5) The partitioning steps design

From the above descriptions, the complete time $T_{i,max}$ is determined by the degree of parallelization, the number of tasks and cores. This paper is aim to solve the minimum MCT problem of the system, and the partition steps are described as follows:

Step1: Code the module in the form of formula (1), initial parameters, such as coupling weight w_c, basic scheduling t_{sch}, basic coupling cost t_{coup}, the limited max ET t_{max}.

step2: for i =1 to k_1

Search the functional cohesion fun_i modules to cluster, and obtain the different functional $task_i$;

Compute the ET t_i of the $task_i$

if $t_i < t_{max}$ //If the ET t_i is larger the limitation of ET, the task should be partitioned in the next step.

goto step 2;

step3: for j =1 to k_2

Divide $task_i$ into subtasks $task_{i,j}$ according to sequential cohesion $seq_{i,j}$;

Compute the ET $t_{i,j}$ of the $task_{i,j}$

if $t_{i,j} < t_{max}$

goto step 3;

step4: for k =1 to k_3

Divide $task_{i,j}$ into subtasks $task_{i,j,k}$ according to sequential cohesion $com_{i,j,k}$;

Compute the ET $t_{i,j,k}$ of the $task_{i,j,k}$

if $t_{i,j,k} < t_{max}$

goto step 4;

step5: for l =1 to k_4

Divide $task_{i,j,k}$ into subtasks $task_{i,j,k,l}$ according to sequential cohesion $tem_{i,j,k,l}$;

Compute the ET $t_{i,j,k,l}$ of the $task_{i,j,k,l}$

if $t_{i,j,k,l} < t_{max}$

goto step 5;

step6: for m =1 to k_5

Divide $task_{i,j,k,l}$ into subtasks $task_{i,j,k,l,m}$ according to sequential cohesion $log_{i,j,k,l,m}$;

Compute the ET $t_{i,j,k,l,m}$ of the $task_{i,j,k,l,m}$

if $t_{i,j,k,l,m} < t_{max}$

goto step 6;

step7: for n =1 to k_6

Divide $task_{i,j,k,l,m}$ into subtasks $task_{i,j,k,l,m,n}$ according to sequential cohesion $coi_{i,j,k,l,m,n}$;

Compute the ET $t_{i,j,k,l,m,n}$ of the $task_{i,j,k,l,m,n}$;

endfor

endfor

endfor

endfor

endfor

end for

Step8: Set the number of cores according to the requirements of the max ECT and the cohesion;

Step9: Calculate the parallelization parameter p_i *, and schedule the subtasks on the cores in the optimal scheduling rule, and compute the complete time of the subtasks on each core in term of formula (8);*

Step10: Search the task of min executing time and the near cohesion task with it, and then execute as following steps:

First, if $h_i > k_i$

If there are the coarse granularity tasks, the tasks are divided into the smaller subtasks.

Second, $h_i \le k_i$

If there are the fine granularity tasks, the tasks are united into the larger tasks.

Third, reschedule the subtasks on the cores in the optimal scheduling rule;

At last, compute the complete time of all the subtasks on each core in term of formula (8)

Step11: Redo step 10 until obtaining the min ECT.

From above steps, the partitioning method is divided into two stages: One is to partition tasks according to cohesion and coupling, and modules can be dispatched as different cohesion tasks to meet the maximum executing time limitation. Another is to partition or combine the subtasks to satisfy the minimum MCT requirement of system.

6) The partitioning method analyze

In this approach, tasks can be partitioned to meet the requirement of the maximum complete time. At step 2, the max executing times of the fine granularity task partition process is not more than $N_{max} = max(n_{fun} \cdot \cdot n_{ord} \cdot n_{inf} \cdot n_{tim} \cdot n_{log} \cdot n_{occ})$. And at the step 11, the algorithm can split or unite the tail tasks, the repeating times do not exceed N_{max} which determined by the max complete time. So the total executing times of the algorithm is less than $2N_{max}$, and the NP-Hard problem is transferred to limited times calculation.

V. THE EXPERIMENTS OF TASK PARTITIONING METHOD

Based on the above analysis, these modules were encoded according to their different cohesion properties, and the ET of these modules is generated randomly. Then different cohesion tasks were obtained in different cohesion stage. Next, we analyze the procedure of the task division based on cohesion and coupling, the ET of tasks, load balance through a series of experiments.

To facilitate the analysis, we assume that basic scheduling time $t_{sch} = 1$, the basic cost of the cohesion is $t_{coup} = 0.5$, $w_{content}$、w_{common}、$w_{external}$、$w_{control}$、w_{stamp}、w_{data} separately equals to 1、1.2、1.4、1.6、1.8、2.0，and the number of the coupling times between different cohesion tasks is 1. Considering the requirements for the system efficiency, load balancing and schedule flexibility, the executing time should be less than 24 times of the schedule time and more than 8 times of the schedule time. The conditions for generating task code separately are: the function cohesion is random number from 1 to 4, other cohesion is random number from 1 to 4, the executing time that each module in the same core is assumed as random data between 2 and 10. The module attribute codes and executing time are shown in TABLE I. The cohesion codes in TABLE I represent the possible classification of the cohesion in according with the equation (1), the [1,5,3,1,6] in m1 row separately represents m1 belongs to the functional cohesion 1, the sequential cohesion 5, the communicational cohesion 3, the temporal cohesion 3 or the logical cohesion 6.

The follows will analyze the change of the coupling cost, task executing time and the scheduling cost in different partitioning stages based on the data in TABLE I.

TABLE I.
THE COHESION CODE AND THE ET OF TEST MODULES

No	Cohesion code						ET	No	Cohesion code						ET	No	Cohesion code						ET
m1	1	5	3	3	1	6	6	m21	2	0	4	2	5	2	5	m41	3	3	4	0	6	6	3
m2	2	3	6	2	6	5	4	m22	4	2	1	0	1	3	10	m42	1	6	5	1	4	0	4
m3	4	4	1	2	1	6	7	m23	4	3	4	5	5	6	2	m43	4	4	1	0	0	6	7
m4	2	5	6	5	3	0	2	m24	2	3	3	6	4	1	6	m44	2	2	1	3	0	0	6
m5	2	0	5	5	2	5	7	m25	2	3	3	2	1	6	4	m45	1	3	4	0	0	3	2
m6	3	6	3	6	4	6	8	m26	4	4	3	1	4	4	9	m46	3	4	5	3	1	1	2
m7	1	6	4	3	6	1	10	m27	3	0	6	6	4	6	5	m47	2	1	6	0	3	5	4
m8	4	4	6	0	5	0	8	m28	2	6	2	0	4	3	8	m48	4	5	3	5	3	1	5
m9	2	1	4	4	4	5	8	m29	2	5	2	6	1	6	2	m49	1	6	6	2	4	6	6
m10	4	6	1	5	0	5	5	m30	4	5	2	4	4	2	5	m50	3	4	3	5	4	0	10
m11	3	3	2	6	0	4	7	m31	3	5	2	5	1	3	6	m51	1	0	5	0	3	3	7
m12	1	5	1	1	0	5	9	m32	2	4	2	1	3	1	5	m52	2	5	5	4	0	6	4
m13	4	2	3	4	2	4	4	m33	3	1	2	3	4	3	7	m53	3	4	1	6	1	6	6
m14	3	6	2	3	5	6	2	m34	2	1	3	0	4	2	7	m54	1	4	2	2	2	2	7
m15	4	0	3	6	6	1	7	m35	3	4	5	5	4	4	3	m55	4	0	3	6	1	3	3
m16	1	3	4	4	0	2	8	m36	1	3	2	3	6	4	10	m56	2	2	5	0	5	3	9
m17	3	2	0	1	5	1	9	m37	3	0	0	2	0	6	8	m57	3	2	3	0	2	1	10
m18	2	6	2	3	4	0	6	m38	2	0	3	2	2	1	9	m58	4	1	3	1	4	0	7
m19	2	2	4	1	3	0	5	m39	4	1	3	6	5	0	9	m59	2	5	3	6	6	3	2
m20	1	4	1	6	1	5	6	m40	4	2	0	2	2	3	8	m60	1	5	0	0	3	5	9

In term of the present partitioning algorithm, the task modules all satisfy the requirement of the minimum MCT, after the functional, sequential, communicational cohesion clustering, there are some fine granularity tasks, which executing efficiency is lowered by the scheduling cost and the coupling cost. So it is necessary to combine those into some new larger tasks to reduce the scheduling and coupling cost.

During the partitioning process of the four stages, the changes of the executing time of tasks are showed in Fig.2. In the division stages, with the increasing of the tasks' number, the max executing time of tasks is decreasing. But the granularity of the task varies differently. After the combination stage, the granularity becomes more even.

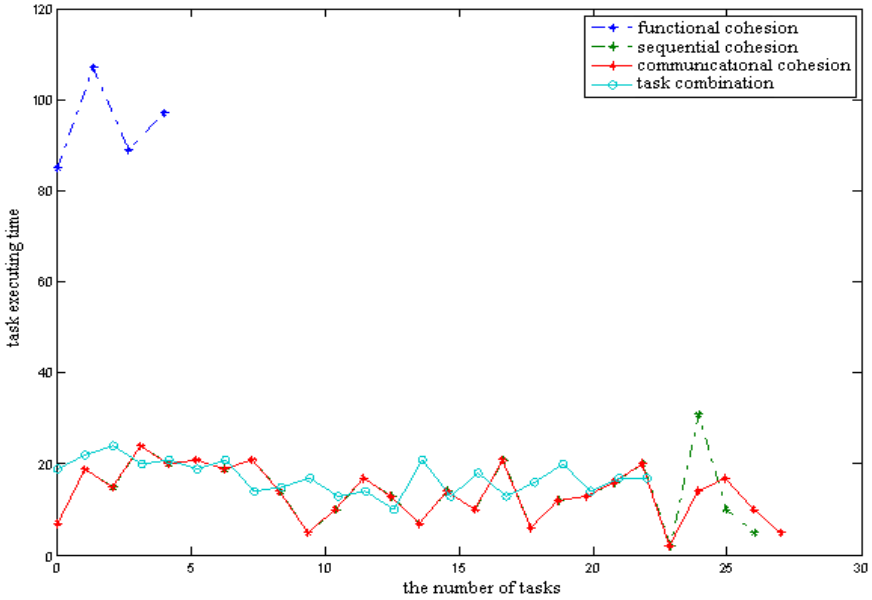

Figure2. The changes of the minimal executing time during the partitioning process

Supposed that the numbers of the cores that can run functional cohesion tasks are 3, 3, 4, 2 respectively, and the average scheduling cost, the coupling cost, the average executing time can be calculated according to formula (8) in each partitioning stage. The computing results are showed in TABLE II.

TABLE II.
THE ET LIST IN EACH STAGE

Partition stage	Number of tasks	$T'_{u,sch}$	$T'_{u,coup}$	T'_u	$T'_{i,max}$
0	1	0	0	288	288
1	4	1	1.5	95.5	108.3
2	5	1.7	3.7	17	38.9
	7	2.3	7.7	15.3	46.6
	7	1.75	5.8	12.7	34
	7	3.5	11.55	13.86	66.1
3	5	1.7	3.7	17	38.9
	7	2.3	7.7	38.1	46.6
	7	1.75	5.8	26.1	34
	8	4	15.45	12.1	70
4	4	1.3	2.3	21.25	33.8
	6	2	5.5	17.8	44.4
	6	1.5	4.1	14.8	31.5
	6	3	8.25	16.3	60.9

In the stage 1 and 2, with the increasing of the number of tasks, the scheduling and coupling cost is increasing, and the total cost of task is also rising, and the executing efficiency is decreasing, but the max complete time is reduced. In stage 3, the more tasks are subdivided, both the costs for scheduling and coupling and the max complete time all increase. When tasks are combined in stage 4, the number of tasks is reduced, and the scheduling and coupling cost also is reduced. As a result, the max complete time is reduced.

To validate the accuracy of the formula (7), the compare experiments between the actual complete time and the evaluated complete time are done based on the data in tab.1. The actual time is computed by supposed that tasks are dispatched according to the optimal scheduling rule. The minimal executing time of each core in each stage shows in TABLE III.

TABLE III
THE MINIMAL ET OF EACH CORE IN EACH STAGE

Partition stage	1	2	3	4
The ECT on the first class cores	85	34	34	39
		27	27	22
		24	24	24
The ECT on the 2nd class cores	107	35	35	35
		36	36	36
		36	36	36
The ECT on the 3rd class cores	89	23	23	26
		21	21	24
		21	21	21
The ECT on the 4th class cores	97	24	24	18
		48	48	48
		49	49	49

And then, the actual complete time can be computed in term of formula (3). The error between the actual complete time and the evaluated complete time is calculated as formula (9), the compare results are showed in Tab.4.

$$Error = \frac{\textit{the actual complete time - the evaluated complete time}}{\textit{the actual complete time}} \times 100\% \quad (9)$$

TABLE IV shows that the most time is nearly equal between evaluated value and optimal scheduled value. Whereas, there is an exception in the last stage, which results from the bad load-balancing owing to the fewer tasks. Therefore, the evaluating method would work well if the parallel parameter p_i should less than 1, it is also the basic requirement of system. So the formula (8) is feasible for estimating the completing time in the partitioning process.

TABLE IV.
THE COMPARE BETWEEN THE ACTUAL COMPLETE TIME AND THEEVALUATED COMPLETE TIME

Partition stage	Task number	$T'_{i,max}$	$T_{i,max}$	*Error*
0	1	288	288	0
1	4	108.3	108.3	0
2	26	38.9	40.2	3.2%
		46.6	46.2	-0.9%
		34	33.2	-2.4%
		66.1	66.4	0.5%
3	27	38.9	40.2	3.2%
		46.6	46.2	-0.9%
		34	33.2	-2.4%
		70	72.8	3.8%
4	22	33.8	40.2	17.6%
		44.4	43.4	-2.3%
		31.5	33.1	4.8%
		60.9	62	1.8%

Meanwhile, as the test results showing, the task completing time will be reduced if more core are provided to execute the task of cohesion code 4. So the number of cores, the system with the restraint for min-max completing time, must be considered.

VI. CONCLUSIONS

This paper analyzes the relation between the cohesion and coupling of tasks on the heterogeneous multi-core processor, designs the cohesion code for task partitioning model, and proposes a novel task partitioning approach to obtain the highest executing efficiency and the well balancing load. The experimental results show the method is effective to reduce the partitioning time and easy to realize. The method also provides a reference for scheduling algorithm design.

In addition, the same cohesion tasks are supposed to be dispatched and run on the same core. But in the practice scheduling process, those tasks can also be allocated on different core to reduce the max complete time according to load balance.

REFERENCES

[1] Bin Li, Xiaowei Zhang, Jun Wu, Junwu Zhu. Task Schedulable Problem and Maximum Scheduling Problem in a Multi-agent System [J]. *Journal of Software,* 2011, 6(11): 2225-2231.

[2] Yuanlong Chen, Dong Li, Peijun Ma. Implementation of Multi-objective Evolutionary Algorithm for Task Scheduling in Heterogeneous Distributed Systems[J]. . *Journal of Software*, 2012, 7(6): 1367-1374.

[3] Sathish Gopalakrishnan, Marco Caccamo. Task partitioning with replication upon heterogeneous multiprocessor systems[C]. *RTAS '06 Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp.199-207, April 2006.

[4] Nathan Fisher, James H. Anderson, Sanjoy K. Baruah. Task partitioning upon memory-constrained multiproce -ssors[C]. *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2005),* Hong Kong, China. pp.416-421, August 2005.

[5] Sanjoy K. Baruah, Partitioning real-time tasks among heterogeneous multiprocessors[C]. *Proceedings of the 2004 International Conference on Parallel Processing* (ICPP'04): 467-474.

[6] S. Arash Ostadzadeh, Roel J. Meeuws, Kamana Sigdel, Koen Bertels. A Multipurpose Clustering Algorithm for Task Partitioning in Multicore Reconfigurable Systems [C]. *International Conference on Complex, Intelligent and Software Intensive Systems-CISIS*, 2009, pp.663-668.

[7] ZHENG Kai. Task Partition Research for Data in Heterogeneous multi-core Processor Simulator [D]. *Shanghai Jiao Tong University,* 2009.

[8] Wu, M. and X. Sun. Memory Conscious Task Partition and Scheduling in Grid Environments[C]. *Proceedings in the 5th IEEE/ACM International Workshop on Grid Computing*. 2004: IEEE Computer Society.

[9] Lastovetsky, A. and R. Reddy, Data partitioning for multiprocessors with memory heterogeneity and memory constraints[J]. *Sci. Program*. 2005, 13(2): 93-112.

[10] Seungyong Oh, Jungsoo Kim, Seonpil Kim , Chong-Min Kyung.Task Partitioning Algorithm for Intra-Task Dynamic Voltage Scaling[R]. *International Symposium on Circuits and Systems* (ISCAS 2008):1228-1231.

[11] YAO Fang-wu1,LU Zhao-cai1.Immune Algorithm of Alterable Coefficient of Parameters Driven Partitioning for Dynamically Reconfigurable System[J]，*Computer Technology and Development*, 2009,19(7):52-55.

[12] G. Wang, W. Gong, R. Kastner. Application Partitioning on Programmable Platforms Using the Ant Colony Optimization[J]. *Journal of Embedded Computing*, 2006, 2(1):119-136.

[13] T. Wiangtong, P. Y. K. Cheung, and W. Luk. Comparing Three Heuristic Search Methods for Functional Partitioning in HardwareSoftware Codesign[J]. *Design Automation for Embedded Systems*, 2002, 6(4):425-449.

[14] T. Wiangtong, P. Y. K. Cheung, and W. Luk. Tabu Search with Intensification Strategy for Functional Partitioning in Hardware-Software Codesign[C]. In *Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2002, pp.297-298.

[15] B. B. AGARWAL,S. P. TAYAL,M. GUPTA. Software engineering & testing an introduction[M]. *America: Jones and Bartlett Publishers*, 2008.

Jianchun Jiang, Doctor, Associate professor. He is engaged in computer control technology, embedded system technologies and applications, intelligent control and other areas, especially do well in researching the embedded software technology, unicore and multi-core processor embedded operating system.

He has achieved two national invention patents, edited one teaching material and co-edited two teaching materials, got three national software copyright registrations. And managed and participated in over eight provincial and ministerial scientific research projects. Among them, one is the national key science and technology subject, three of them are the national "863 Plan" projects and two of them are key research projects of Chongqing science and technology.